WebTech Coursework 2 Report

Luke Reeder
40277803@napier.ac.uk
Edinburgh Napier University - Web Tech (SET08101)

1 Introduction

The assignment given to myself for the coursework was to implement a working blog platform. This was to be done by using HTML, CSS, JS and Node.js. My site is one that includes Node[1] and MongoDB[2] for services to run the server and the database used to store and retrieve the posts from a user. It includes the functionality to make new posts to the database, view all posts in a list, remove posts and also amend any posts in the title or posts themselves. For the background reading for the coursework, I used the github repositories for the node modules that I used throughout the coursework to learn more about their functions and their use cases, along with how to set them up within the server.

2 Software Design

Before I started with the implementation of the code, I planned a rough design for the site layout and functions. I decided that I wanted to make the site a simplistic design as to not be overly cluttered and messy, as well as having clearly laid out navigation between pages and their functions. I knew I wanted to keep the design a light coloured with having a navigation bar at the top of the screen being bordered off. For the view posts page, I wanted to make sure that the the posts were arranged in a list in the center of the page, with the titles being slightly larger font and bold to draw attention to it. Below the text would be clearly laid out using a plain but easily readable font selection. After both the post tite and the post body, I wanted to have the buttons clearly laid out beneath the post to allow the user to see which buttons related to which post. The buttons were made as a dark green colour to clearly stand out from the light blue background. I also wanted to have the background of the list to be slightly lighter than the actual page background to be able to differentiate between the two sections.

For the New Post page, I wanted to keep the design choices consistent across the site, so once again I used a slightly lighter coloured background for the section, as well as having the corners rounded slightly to give it a softer feeling. The submit button was again a dark green colour to stand out from the rest of the page.

In the navigation bar, I made the background slightly lighter than the page to keep the style similar but consistent and pleasing to the eye. I also added a function to change the selections colour while hovering over each button to clearly indicate which page you'd like to navigate to with ease.

3 Implementation

For the site I had implemented the user functionality to be able to add, amend and remove posts from the blog platform. This was achieved by setting up the server using Node.Js as the service 1 and using Atom as a text editor to write the code. I also used MongoDb to store the posts on a server to have the posts be persistent. I installed a variety of node modules to allow the different functions. These were Express, MongoDb, MongoJs[4], NodeMon[6], Body-Parser[3], Ejs[7] and Express Validator[5].

Express was used to write the majority of the actual server side functions, such as hosting the pages and having the server run the add/amend/delete functions. MongoDB and MongoJs were used to handle and store the posts, as well as having an ORM to allow the individual posts to be accessed with ease to allow for editing. Nodemon was an extremely helpful module that runs the server and refreshes it on change to the source code. This saved a lot of time when making small changes and for adding new functions. Express validator alongside Body-Parser were used to read in the bodies of text from the user and validate that the post and title were actually filled in rather than blank or just white space. Lastly, Ejs was used to embed the javascript functions onto the server to allow the use of the add 2, amend and delete functions.

4 Critical Evaluation

For the site I implemented the functionality for a user to be able to make a new post, to be able to view all of the current posts, to delete any posts and to amend the post or post title. This functionality was listed as required functions, but as for possible improvements to the site I would make the site more secure. To do this I would make it so that the user has to log in to the site to be able to amend or delete the posts. An additional feature I would add to this would be to allow the user to post anonymously to the site if they so wished by having the user select a username on account registration, then having a separate button that would allow them to post anonymously or using their handle.

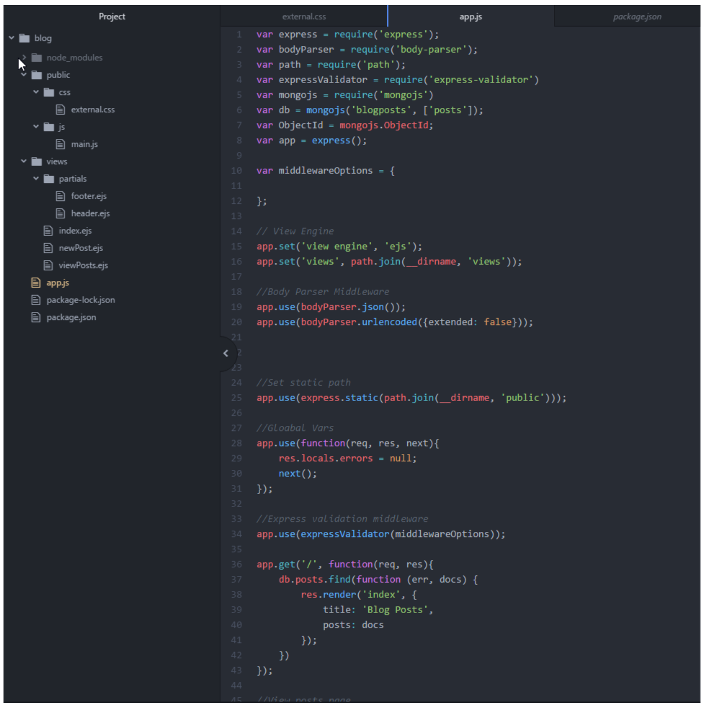
Other possible improvements would be to allow a user to also attach images to their posts rather than just text. Another possible addition would be to add a user page to the site to allow the user to write a brief description about themselves and add any other info they would like to.

5 Personal Evaluation

Throughout this coursework, I learned how to properly implement a functioning Node.JS server along with using various different node modules to help improve functionality and usability for the site as a whole. Such examples include express for the general server side navigation and implementation of user functions, body parser and express validator to be able to pass in bodys of text and validate them so that no posts are left blank and take up space without being relevant. I also learned how to utilize MongoDB to store the posts and retrieve them to have the website be persistent past restarting servers to make adjustments to code or adding additional functions.

Some of the challenges I had while working on this coursework were trying to make sure that the functions between the server and the database were smooth and properly working. Along with silly user errors such as missing dots or semicolons, I had slight problems implementing the delete function due to the objectID not properly referencing. This was solved by inspecting each section and rewriting it line by line to make sure that each part was properly written.

Other slight problems were luckily not as hard to solve, such as having multiple pages hosted on the same server along with proper navigation between pages and functions. To solve this part I tested each button on the page as well as looked at each nav function in the code to double check that every part worked as intended.



```
var express = require('express');
var bodyParser = require('body-parser');
var path = require('path');
var expressValidator = require('express-validator')
var mongojs = require('mongojs')
var db = mongojs('blogposts', ['posts']);
var ObjectId = mongojs.ObjectId;
var app = express();

var middlewareOptions = {

};

// View Engine
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

//Body Parser Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));



//Set static path
app.use(express.static(path.join(__dirname, 'public')));

//Gloabal Vars
app.use(function(req, res, next){
    res.locals.errors = null;
    next();
});

//Express validation middleware
app.use(expressValidator(middlewareOptions));

app.get('/', function(req, res){
    db.posts.find(function (err, docs) {
        res.render('index', {
            title: 'Blog Posts',
            posts: docs
        });
    })
});

//View posts page
```

Figure 1: Server App

6 References

The additional resources used for this coursework were Node.js and MongoDb.

References

References

[1] Node.Js `https://nodejs.org/en/`

[2] MongoDb `https://www.mongodb.com/`

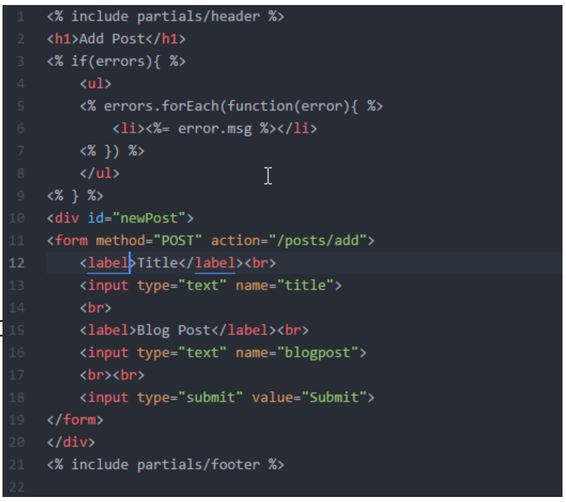
[3] Body-Parser `https://github.com/expressjs/body-parser`

[4] MongoJs `https://github.com/mafintosh/mongojs`

[5] Express Validator `https://github.com/ctavan/express-va`

[6] Nodemon `https://github.com/remy/nodemon`

[7] Embedded Javascript `https://github.com/mde/ejs`



```
<% include partials/header %>
<h1>Add Post</h1>
<% if(errors){ %>
    <ul>
    <% errors.forEach(function(error){ %>
        <li><%= error.msg %></li>
    <% }) %>
    </ul>
<% } %>
<div id="newPost">
<form method="POST" action="/posts/add">
    <label>Title</label><br>
    <input type="text" name="title">
    <br>
    <label>Blog Post</label><br>
    <input type="text" name="blogpost">
    <br><br>
    <input type="submit" value="Submit">
</form>
</div>
<% include partials/footer %>
```

Figure 2: New Post Code